Confidence and security start with understanding what software is being used.

What is in your software bill of materials (SBOM)?
Software Bill of Materials

A software bill of materials (software BOM) is a list of components in a piece of software. Software vendors often create products by assembling open source and commercial software components. The software BOM describes the components in a product.\[1\][2] It is analogous to a list of ingredients on food packaging.


source: https://www.pinterest.ca/pin/278660295677291647/
BOM Documentation (1)

BOM: „Bill of Material“
- It is a general question what is in the delivery
- Understand the nature of the delivery (How much OSS?)
- Understand potential issues (IP)
- How else to ensure license compliance?
- Basics of supply chain issues actually apply also to software
- Software Package Data Exchange (SPDX) specifies one implementation how to express a BOM of a software package [1]

[1] https://spdx.org/

BOM Documentation (2)

Bill of material can be general obligation, for example at:
- USA: Cyber Supply Chain Management and Transparency Act of 2014
  - Obliged to report service disturbances
  - Obliged to implement information security
  - Requires knowledge about BOM

A product’s components can be created from open source and proprietary software packages.

Modern open source projects are an interwoven set of multiple dependencies on other projects with multiple versions of source and licenses applying.
Challenge: Accurately summarizing the information

Different programming languages impose different standard ways of handling third party dependencies.

› Java (e.g. ONAP): retrieved at build time
› Go (e.g. Kubernetes, Hyperledger Fabric): checked into source code repository

Adds complexity when thinking about what components are “in” the project code.
Sharing SBOMs: What Information is Significant?

Information for Developers

List of packages being used.

Includes name, version number, checksums, download location, source location, license information, build and run dependencies, vulnerability identifiers etc.

Tooling is needed to build up a product and track components
Sharing SBOMs: What Information is Significant?

Information for Management

Summary of license and vulnerability findings, tailored for discussion with legal counsel and executives for risk management.

May include expert recommendations, and dynamic generated information to assist with evaluating requests.
Software “ingredient” management has not been standardized:

• Varies by language, distro, repository, company...
• Incomplete or inconsistent license notices and security info
• Time-consuming to manually examine files
• Time-consuming to maintain scanning processes

Most developers want to build software, not spend time dealing with packaging, labeling and distribution.
What’s Done Today?

› Run scans on incoming project codebases and patches
› Analyze license texts, notices and security references to determine if can use internally, compose with other components, distribute
› Produce summary reports of “key information” for builds, releases, patches, archives, etc. to share with consumers
› Try to apply “best practices” as understood within projects, communities, organizations and companies
“Key Information”

› Varies from project to project
› Varies from tool to tool
› Varies from company to company
› Varies from community to community

How do we agree on these “Nutrition Facts” we want to keep with our “Ingredients”? 
Is there a reference “open source tools” based workflow? not yet...
Many Tools and Approaches...

› Approaches:
  › **Processes & Practices:** OpenChain, REUSE Initiative, CII Best Practices, OWASP
  › **Source & Metadata Repositories:** Github, Gitlab, Software Heritage, Libraries.io, npm1K.org, ClearlyDefined, ...

› Tooling:
  › **Commercial Tools:** <insert favorite vendor here> ...
  › **Open Source:** FOSSology, SPDX-Tools, Quartermaster, SW360, OSS Review Toolkit, ScanCode, LicenseFinder, TERN, CopyrightReviewTools, ...

What key “Nutrition Facts” do we need to keep with our “Ingredients”?
But we do have a language to “exchange” facts

Software Package Data Exchange® (SPDX®)
is an open standard for communicating software bill of material information (including components, licenses, copyrights, and security references).
Openly Created SBOM format for the “Nutrition Facts”

The Software Package Data Exchange (SPDX®) Specification Version 2.1.1

Copyright © 2010–2018 Linux Foundation and its Contributors. This work is licensed under the Creative Commons Attribution License 3.0 Unported (CC-BY-3.0) reproduced in its entirety in Appendix VII herein. All other rights are expressly reserved.


https://spdx.github.io/spdx-spec/

Used to communicate software identification, license and security information in standardized, machine-readable formats

SPDX files can be produced from source code scans or builds, curated and annotated by reviewers, and shared between organizations

Based on 8 years of analysis of use cases, incorporating input from industry experts in packaging, licensing and security
Use Case: Generating SBOMs for Incoming Open Source package?
Tools: FOSSology

Server based framework used to scan a codebase for licenses, copyright and export statements

Performs textual analysis and regular expression scanning to identify likely license notices and references

Manual review results to remove false positives and investigate unusual findings, can be retained between versions

https://www.fossology.org/
Tools: ScanCode

ScanCode

Is able to generate SPDX documents!

In the current code** (2.9b1) you can use the options:

--output-spdx-rdf FILE Write scan output as SPDX RDF to FILE.
--output-spdx-tv FILE Write scan output as SPDX Tag/Value to FILE.

Both options can be used, and create both files from a single scan.

** 2.2.1 had the capability using --format spdx-rdf and --format spdx-tv

Need to manually review output to remove false positives and investigate unusual findings

https://github.com/nexB/scancode-toolkit
Tools: OSS Review Toolkit

An analyzes the source code for dependencies, downloads the source code of the dependencies, scans all source code for license information, and summarizing the results.

ORT is a suite of different tools that are designed as libraries (for programmatic use) with a minimal command line interface (for scripted use).

https://github.com/heremaps/oss-review-toolkit
Case Study: Applying a update from Open Source (patch, new version, etc.)
Tools

**FOSSology**: Import update into server, reuse existing results when no change in code, scan and clear any changes. Able to import prior SPDX documents and merge.

**ORT**: Incorporate updates and analysis.

**ScanCode**: Need to redo ScanCode analysis (AboutCode framework does not work with SPDX)
Case Study: Creating Executables with SBOMs
Tools

**FOSSology**: Run after build. Able to import prior SPDX documents and merge between changes.

**ORT**: Incorporate updates and analysis.

**ScanCode**: Can add into CI/CD flow, but need to inspect after. Diff with prior, etc.
Tools: Quartermaster (QMSTR)

Open Source license compliance audit as part of a modern DevOps cycle.

Workflow engine integrates existing scanning and reporting tools, and integrates into CI/CD processes.
Tools: SPDX Maven Plugin

Include all build dependencies and source file information it can detect (not a license and copyright scanner)

Generates a valid SPDX file from ANY POM file.

https://github.com/spdx/spdx-maven-plugin
Case Study: Distributing SBOM with an Executable
Tools

**FOSSology**: Exports SPDX documents that describe executable, can link to source repository SPDX document that describes sources. Multiple Report formats.

**SPDX Maven Plugin**: Uploads automatically SPDX document to Maven

**ORT**: *Documenter* - Generates the outcome of the review, e.g. Open Source notices and annotated SPDX files that can be included into your distribution.
Case Study: Publishing SBOM to Share
How? Emerging Best Practices

Core Infrastructure Initiative (Security & Transparency)

REUSE (Artifact organization)

OpenChain (Process between organizations)

SPARTS (Distributed ledger for logging & tracking in supply chain)
Best Practices: Transparency → Security

CII Best Practices

To achieve Gold badge status, accurate copyright and licensing information is required.

see:
Best Practices: Upstream Workflows

**REUSE** - developed by FSFE describes best practices for describing licensing information in open source software and making it suitable for automation.

See: [https://reuse.software/](https://reuse.software/)

Practices: [https://reuse.software/practices/2.0/](https://reuse.software/practices/2.0/)

Overview: [https://reuse.software/reuse/reuse-presentation.pdf](https://reuse.software/reuse/reuse-presentation.pdf)
Best Practices: Between Organizations

OpenChain project documents the processes to build trust between members of a software supply chain using open source software.

The OpenChain Project builds trust in open source by making open source license compliance simpler and more consistent. The OpenChain Specification defines a core set of requirements every quality compliance program must satisfy. The OpenChain Curriculum provides the educational foundation for open source processes and solutions, whilst meeting a key requirement of the OpenChain Specification. OpenChain Conformance allows organizations to display their adherence to these requirements. The result is that open source license compliance becomes more predictable, understandable and efficient for participants of the software supply chain.
Best Practices: Between Organizations

**Software Parts Ledger** - utilizes Blockchain to manage open source across the supply chain. Utilizes Hyperledger Sawtooth Platform & SPDX based BOM to conform to OpenChain best practices.

See: [https://github.com/Wind-River/sparts](https://github.com/Wind-River/sparts)

Accepted 2018/3 into Hyperledger Labs - [https://github.com/hyperledger-labs/hyperledger-labs.git hub.io/blob/master/labs/SParts.md](https://github.com/hyperledger-labs/hyperledger-labs.git hub.io/blob/master/labs/SParts.md)
So, how to generate an SBOM for Linaro Deliverables?
Fishbowl Discussion: Linaro initial Experience

- Feels like the wild west stepping into the SPDX world
- Linaro has worked with FOSSology and ScanCode
  - FOSSology is heavy weight, requires a server
  - ScanCode appears to be simpler, but cannot remember amendments to the generated output making diff difficult
- Lack of projects to use as a reference
- Integration with GitHub
Why create an SPDX document?

Can be very descriptive

• Accurate manifest of the software contents (copyrights, licensing, dependencies, provenance, CPE, …)

Can be easily translated to human readable forms

• Spreadsheet, grep’able text file (tag:value), viewers

Share data between open source and commercial tools

HELP the downstream user understand and comply with the upstream developer’s licensing expectations!
What is an SPDX Document?

### Package Information

---

PackageName: time-1.7.tar.gz
PackageName: time-1.7.tar.gz
PackageDownloadLocation: NOASSERTION
PackageVerificationCode: dd5cf0b17b9ef4284c6c22471b277dc7beac407c
PackageChecksum: SHA1: dde0c28c742b696736933f3e763320680356cc6a
PackageLicenseConcluded: GPL-2.0+
PackageLicenseDeclared: GPL-2.0+
PackageLicenseInfoFromFiles: GPL-2.0
PackageLicenseInfoFromFiles: MIT
PackageLicenseInfoFromFiles: LicenseRef-1
PackageLicenseInfoFromFiles: LicenseRef-2
PackageLicenseInfoFromFiles: LicenseRef-3
PackageCopyrightText: NOASSERTION

---

RDF/XML
What makes up an SPDX Document?

SPDX v2.1 Document contains:

- Document Creation Information
- Package Information
- File Information
- Snippet Information
- Other Licensing Information
- Relationships
- Annotations
Only subset of fields are mandatory

**Document Creation Information**
- 2.1 SPDX Version.
- 2.2 Data License
- 2.3 SPDX Identifier
- 2.4 Document Name
- 2.5 SPDX Document Namespace
- 2.8 Creator
- 2.9 Created

**Package Information**
- 3.1 Package Name
- 3.2 Package SPDX Identifier
- 3.7 Package Download Location
- 3.9 Package Verification Code
- 3.13 Concluded License
- 3.14 All Licenses Information from Files
- 3.15 Declared License
- 3.17 Copyright Text

**File Information**
- 4.1 File Name
- 4.2 File SPDX Identifier
- 4.4 File Checksum
- 4.5 Concluded License
- 4.6 License Information in File
- 4.8 Copyright Text

1 per document
1 per package in document
1 per file in each package
Creating an SPDX document

Requires tooling as the package verification code is generated by file `checksums(aka hashes)` and the total number of files per useful package.